

Esercizio 1

Si discuta l'accesso in memoria in un'architettura di Von Neumann (NB: ci sono diversi tipi di memoria).

Correzione

Occorreva discutere l'ACCESSO in memoria, non i diversi tipi di memoria in generale. Esempio: RAM (indirizzamento, accesso casuale quindi tempo di accesso costante, trasferimento dei dati su bus indirizzo, dati e controllo) e idem per REGISTRI (accesso più veloce, non serve trasferimento perché già presenti nella CPU vicino ad ALU), MEMORIE DI MASSA (es. hard disk: dischi settori, tracce, tempo di latenza, di trasferimento, etc.). Importante anche il FileSystem.

Esercizio 2

Si commenti la relazione tra l'algebra di Boole e l'informatica.

Correzione La relazione INFO/BOOLE sta nel dualismo: 0/F 1/V +/OR */AND

Esercizio 3

Analizzare attentamente il seguente listato di codice e rispondere alle seguenti domande:

1. Spiegare l'effetto di ciascuna delle tre funzioni (**f1**, **f2**, **f3**) su una stringa in ingresso.
2. Per ciascuna funzione, spiegare se ed in quali casi la sua invocazione potrebbe causare errori di accesso ad aree protette di memoria (*segmentation fault*).
3. Cosa stampa l'istruzione **printf** alla riga 13?
4. Cosa stampa l'istruzione **printf** alla riga 14?
5. Cosa stampa l'istruzione **printf** alla riga 15?
6. Cosa stampa l'istruzione **printf** alla riga 16?

```
1
2 void f1(char* x) { x[0] = x[strlen(x)]; }
3
4 void f2(char* x) { *x = *(x+2); }
5
6 void f3(char* x) { int a; x[a] = x[0]; }
7
8 int main() {
9     char a[] = "getto"; char b[] = "masso"; char c[] = "letto";
10
11     f1(a); f2(b); f3(c);
12
13     printf("%s\n", a);
14     printf("%s\n", a+1);
15     printf("%s\n", b);
16     printf("%s\n", c);
17
18     return 0;
19 }
```

Correzione La funzione **f1** non causa errori di accesso ad aree protette di memoria (*segmentation fault*) mentre **f2** sì solo se la stringa è più corta di due caratteri. In **f3** dipende dal valore assegnato dal compilatore alla variabile locale "a", che non essendo stata assegnata conterrà un valore casuale, quasi sicuramente un indice per una zona di memoria protetta.

L'istruzione **printf** alla riga 13 stampa la stringa vuota (avendo come primo carattere il terminatore di stringa la **printf** si ferma subito)

L'istruzione **printf** alla riga 14 stampa "etto" perché parte dall'indirizzo "a+1" quindi dal secondo elemento dell'array.

L'istruzione **printf** alla riga 15 stampa "sasso" perché $*x = *(x+2)$ corrisponde a $x[0] = x[2]$.

È impossibile sapere con certezza cosa stamperà l'istruzione **printf** alla riga 16. Es. stampa "letto" se "a" fosse inizializzata a 0, ma poiché a corrisponde ad un valore casuale molto probabilmente non nel range (0, lunghezza stringa) darà un *segmentation fault*.